Technical Cybersecurity

gdb

gdb

GNU DeBugger

- binary level debugger
- Used with the GNU compiler collection
  - Can use with other executables too, like LLVM
  - Though LLVM has LLDB

UNIX DEBUGGERS

- GDB, usually used with Linux distress
- LLDB, usually used with MacOS
- …but you can use either with the other

Command Line

GUIs

- DDD
- …well, that's kinda it

Usually used via CLI

- Very powerful, not to difficult to learn
- Cheatsheets online
  - https://darkdust.net/files/GDB%20Cheat%20Sheet.pdf
  - https://cs.brown.edu/courses/cs033/docs/guides/gdb.pdf
  - https://gist.github.com/rkubik/b96c23bd8ed58333de37f2b8cd052c30
  - Google: gdb cheatsheet

Let’s use it!

GDB IS WORDY

- -q fixes that
- set an alias:
  - **$ alias gdb=“gdb -q”**
  - place in ~/.bashrc

WORDS MEAN WHAT?

- Warranty, configuration info, bug reporting, manual location, etc.
- one and done!

```
cclamb@ubuntu:~/Work/abi-playground $ gdb f2
GNU gdb (Ubuntu 8.1-0ubuntu3) 8.1.0.20180409-git
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://g
This is free software: you are free to change and re
There is NO WARRANTY, to the extent permitted by law
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resource
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related t
Reading symbols from f2...done.
(gdb) quit
cclamb@ubuntu:~/Work/abi-playground $ gdb -q f2
Reading symbols from f2...done.
(gdb) quit
cclamb@ubuntu:~/Work/abi-playground $ alias gdb="gdb
cclamb@ubuntu:~/Work/abi-playground $ gdb f2
Reading symbols from f2...done.
(gdb)
```

```
============== B E G I N N I N G

        _start:
3b0             xor         ebp, ebp
3b2             mov         r9, rdx
3b5             pop         rsi
3b6             mov         rdx, rsp
3b9             and         rsp, 0xff
3bd             push        rax
3be             push        rsp
3bf             mov         r8, __lib
3c6             mov         rcx, __li
3cd             mov         rdi, main
3d4             call        qword [0x
3da             hlt
              ; endp
3db             align       32
--------------------------------------

============== B E G I N N I N G
```

```
ubuntu:~/Work/abi-playground $ gdb f2
 symbols from f2...done.
nfo func
ined functions:

nction2.c:
ll(void);
ll2(void);
n(int, char **);

ugging symbols:
00000400390  _init
000004003b0  _start
000004003e0  _dl_relocate_static_pie
000004003f0  deregister_tm_clones
00000400420  register_tm_clones
00000400460  __do_global_dtors_aux
00000400490  frame_dummy
000004004e0  __libc_csu_init
00000400550  __libc_csu_fini
00000400554  _fini
reak _start
int 1 at 0x4003b0

g program: /home/cclamb/Work/abi-playground/f2

int 1, 0x00000000004003b0 in _start ()
isas
 assembler code for function _start:
0000000004003b0 <+0>:     xor    %ebp,%ebp
0000000004003b2 <+2>:     mov    %rdx,%r9
0000000004003b5 <+5>:     pop    %rsi
0000000004003b6 <+6>:     mov    %rsp,%rdx
0000000004003b9 <+9>:     and    $0xfffffffffffffff0,%rsp
0000000004003bd <+13>:    push   %rax
0000000004003be <+14>:    push   %rsp
0000000004003bf <+15>:    mov    $0x400550,%r8
0000000004003c6 <+22>:    mov    $0x4004e0,%rcx
0000000004003cd <+29>:    mov    $0x4004bc,%rdi
0000000004003d4 <+36>:    callq  *0x200c16(%rip)        # 0
0000000004003da <+42>:    hlt
assembler dump.
```

```
============== B E G I N N I N G

        _start:
3b0         xor         ebp, ebp
3b2         mov         r9, rdx
3b5         pop         rsi
3b6         mov         rdx, rsp
3b9         and         rsp, 0xf
3bd         push        rax
3be         push        rsp
3bf         mov         r8, __li
3c6         mov         rcx, __li
3cd         mov         rdi, main
3d4         call        qword [0x
3da         hlt
            ; endp
3db         a
---------------------------------

============== B E G I N N I N G
```

Intel

```
ubuntu:~/Work/abi-playground $ gdb f2
 symbols from f2...done.
nfo func
ined functions:

nction2.c:
ll(void);
ll2(void);
n(int, char **);

ugging symbols:
00000400390  _init
000004003b0  _start
000004003e0  _dl_relocate_static_pie
000004003f0  deregiste
00000400420  register_
00000400460  __do_glob
00000400490  frame_dummy
000004004e0  __libc_csu_init
00000400550  __libc_csu_fini
00000400554  _fini
reak _start
int 1 at 0x4003b0

g program: /home/cclamb/Work/abi-playground/f2

int 1, 0x00000000004003b0 in _start ()
isas
 assembler code for function _start:
0000000004003b0 <+0>:     xor    %ebp,%ebp
0000000004003b2 <+2>:     mov    %rdx,%r9
0000000004003b5 <+5>:     pop    %rsi
                                 p,%rdx
                                 fffffffffffffff0,%rsp
                                 ax
0000000004003be <+14>:    push   %rsp
0000000004003bf <+15>:    mov    $0x400550,%r8
0000000004003c6 <+22>:    mov    $0x4004e0,%rcx
0000000004003cd <+29>:    mov    $0x4004bc,%rdi
0000000004003d4 <+36>:    callq  *0x200c16(%rip)        # 6
0000000004003da <+42>:    hlt
assembler dump.
```

AT&T

(gdb) set disassembly-flavor intel

.gdbinit

Place in home directory

- Allows you to insert common commands (like setting the disassembly flavor)
  - **$ echo “set disassembly-flavor intel” > ~/.gdbinit**
- You can also use an init file with common commands
  - use the *-x filename* or *-ex command*
  - Your init file won’t be read when you do this
  - Good for common stuff you always do

```
cclamb@ubuntu:~/Work/abi-playground $ cat gdbinit
set disassembly-flavor intel
b _start
r
cclamb@ubuntu:~/Work/abi-playground $ gdb -x ./gdbinit f2
Reading symbols from f2...done.
Breakpoint 1 at 0x4003b0

Breakpoint 1, 0x00000000004003b0 in _start ()
(gdb) disas
Dump of assembler code for function _start:
=> 0x00000000004003b0 <+0>:     xor    ebp,ebp
   0x00000000004003b2 <+2>:     mov    r9,rdx
   0x00000000004003b5 <+5>:     pop    rsi
   0x00000000004003b6 <+6>:     mov    rdx,rsp
   0x00000000004003b9 <+9>:     and    rsp,0xfffffffffffffff0
   0x00000000004003bd <+13>:    push   rax
   0x00000000004003be <+14>:    push   rsp
   0x00000000004003bf <+15>:    mov    r8,0x400550
   0x00000000004003c6 <+22>:    mov    rcx,0x4004e0
   0x00000000004003cd <+29>:    mov    rdi,0x4004bc
   0x00000000004003d4 <+36>:    call   QWORD PTR [rip+0x200c16]        # 0x600ff0
   0x00000000004003da <+42>:    hlt
End of assembler dump.
(gdb)
```

Using -x

Let’s start looking at f2.